Problem F: Identifying Comments

Achilles: Hey, Mr. T. How is it going? Are you heading home?

Tortoise: Hi, Achilles. Yes, I just had another meeting with Aunt Hillary. Do you remember the program I wrote for her and her beauty salon?

Achilles: Sure. But, wait, I thought you had finished that job already.

Tortoise: Oh, yes. It worked perfectly and she liked it a lot. It's just that she now wants to store the source code in some old tape disks, but they are very old and have a very low capacity. So, she wanted to know if there was a way to reduce the size of the files.

Achilles: Ah, you mean, *compressing* them…

Tortoise: Not quite. She wants to keep the source code readable, but shorter. So, we agreed that I would just remove all the comments from my code. That satisfies her, and is not very hard to do. I will even have some fun writing a new program to do that job for me.

Achilles: Did you add a lot of comments to your code? I thought good programmers never wrote comments.

Tortoise: Please, Achilles. I know that you are just jesting, but I am in fact very rigorous with comments. It's not as easy as many people think, you know? It is a subtle balance between not saying things that are evident from the code, and not leaving some parts of the code hard to understand.

Achilles: I know what you mean. Unfortunately many novices don't seem to get that. Or what about people who indent their code incorrectly, or don't indent it at all…

Tortoise: Oh, please, Achilles. Don't get me started with bad indentation. Besides, I don't want to get nightmares tonight. Anyway, as I was saying, I think I can write a short program to remove those comments. I used a programming language with a very simple syntax. It just allows literals, strings and comments, so it shouldn't be too hard to identify those comments, and remove them…

Given the source code of a program in Mr. T's programming language of choice, find the number of comments it has, and the total number of characters used in all comments. Programs in that language are formed by the following elements exclusively:

1. **Literals**. A literal is a sequence of one or more alphanumeric characters (the digits 0..9 and uppercase or lowercase letters from the English alphabet).
2. **Strings**. A string is a sequence of printable characters or single–spaces, that begin and end with a double–quote character (`"`). A string may contain double–quote characters, but they have to be escaped; that is, they have to be preceded by the backslash character (`\`). Since the only whitespace allowed in a string is the single–space, strings cannot span over multiple lines.
3. **Comments**. A comment is a sequence of printable characters, single–spaces or newlines, that begin with the sequence `/*` and end with the sequence `*/`. Comments *can* span over multiple lines, but they cannot contain the sequence `*/`, because that identifies the end of the comment.
4. **Whitespace**. To separate all other elements, the only whitespace allowed are single–spaces and newlines.

Input

Input starts with a positive integer **T**, that denotes the number of test cases. The first line of each test case contains an integer **N**, which denotes the number of lines of code in the program.

N lines follow, each one no longer than 80 characters. You may assume that the code follows the syntax previously described. See the samples for more details.

$T \le 50$; $1 \le N \le 100$

Output

For each test case, print the case number, followed by two integers: the number of comments in the code, and the total number of characters used by all the comments (including the `/*` and `*/` sequences).

Sample Input

```
2
22
/*
 * This is a comment
 * that spans multiple lines
 *
 * I can contain any printable character,
 * like !$%*=?\"/’#^[]()
 */
BEGIN SampleProgram
  VAR i j k  /* variable declaration */
  SET i 10   /* assign ten to i */

  LOOP NZ i  /* loop while i is not zero */
    PRINT "You say \"goodbye\""
    PRINT "And I say \"hello\"!"

    SUB i 1  /* subtract one from i */
  END LOOP

  PRINT "/*******/"
  PRINT "/* Bye */"
  PRINT "/*******/"
END
1
PRINT "Hello World" /* greeting */
```

Output for Sample Input

```
Case 1: 5 228
Case 2: 1 14
```

Notes

A single–space refers to the character with ASCII value 32. A newline refers to the character with ASCII value 10. A double–quote is ASCII value 34. The ASCII value of all printable characters will be in the range $32 \ldots 126$.